ステレオ画像処理を用いた曖昧再利用の評価

津邑公暁[†] 清水雄歩[†] 中島康彦[††]
五島正裕[†] 森眞一郎[†]
北村俊明[†††] 富田眞治[†]

カメラ画像を用いたステレオ画像処理において，最も計算量を要する視差測定部に対し関数再利用を適用することにより，関数再利用の有効性を示す．関数あたりに計算するピクセル値差の多重度を増やす一方，入力のマッチングに寛容さを持たせる曖昧再利用を適用することで，再利用による効果を上げつつ再利用のヒット率も保つことができることを示す．オリジナルプログラムと比較した場合において，曖昧再利用を適用した実装では，最大 90%のサイクル数を削減できた．データ並列度によらず適用可能である再利用で，メディア演算命令に匹敵する効果が得られた．

An Evaluation of Tolerant Function Reuse on Stereo Depth Extraction

Tomoaki Tsumura,[†] Yuho Shimizu,[†] Yasuhiko Nakashima,[††]
Masahiro Goshima,[†] Shin-ichiro Mori,[†] Toshiaki Kitamura[†††]
and Shinji Tomita[†]

This paper describes the effectiveness of tolerant function reuse against the disparity detecting function in stereo depth extraction. The more pixel sets a function processes, the more effective the function reuse is. On the other hand, the less frequently the function will be reused, and the performance will hit its peak. We propose a way using tolerant reuse. It can increase the effectiveness of function reuse and keep the frequency of reuse high. We show the maximum eliminated cycles with our method reaches to 90%. The performance matches implementation with media processing instructions.

1. はじめに

画像処理は膨大な計算量を必要とすることが多く，計算機の性能向上がめざましい近年においても，さらなる速度向上が必要とされている分野である．特に自動監視システムや移動ロボットのためのステレオビジョンといった，実時間処理が必要となるシステムにおいては，その要求は大きなものとなる．

ステレオ画像処理において最も計算時間を要するのは距離計算であり，さらに細かく見るとその距離計算の際に必要となる視差測定である．SONY は，レーザ光を用いた距離測定のためのチップ Entertainment Vision Sensor を開発した[1)]．しかしアクティブセンサシステムはセンサ自身の消費電力を抑えにくいなどの欠点がある．これに対して TYZX などは，パッシブセンサを用いた視差測定専用プロセッサを使用したシステムを開発している[2)]．

このように，現在多くのステレオ画像処理システムでは専用プロセサを用いて距離計算を行っている．しかし，将来的には高速な汎用プロセサを用いてより安価に実現されることが予想されることから，VLIW 型プロセサに搭載されたメディア演算命令を用いることで距離画像生成を高速化しようという試みも行われている[3)]．

我々は，カメラ画像を入力として用いるパッシブセンサ型の距離画像生成プログラムにおいて，関数再利用を適用することにより高速化を行った．最も計算量が必要となる，視差検出時のピクセル値差分を求める関数に対して関数再利用を適用し，さらに関数の入力のマッチングに寛容さを持たせることで，再利用率お

† 京都大学
Kyoto University
†† 京都大学/科学技術振興事業団さきがけ研究 21
Kyoto University/PRESTO, JST
††† 広島市立大学
Hiroshima City University

よびサイクル数削減率を向上させる手法を提案する．速度向上および必要ハードウェア量の両点から，専用メディア演算命令を用いた場合との比較を行う．

以下，2 章ではステレオ画像処理の概要について述べ，3 章で関数再利用のしくみについて述べる．4 章では距離画像生成への関数再利用の適用手法について述べる．5 章で評価結果を示し，6 章で必要ハードウェア量に関して考察する．その後，7 章でまとめを行う．

2. ステレオ画像処理

ステレオ画像処理とは，ある対象を 2 つの異なるカメラから観測して得られる 2 枚の画像から，その対象の**距離画像**を得る手法である．距離画像とは，対象までの距離情報を濃淡により可視化した画像で，距離画像内では距離が近いものは明るく，遠いものは暗く表示される．

一般にステレオ画像処理は，画像のノイズ除去/ぼかしなどの前処理を行ったのち，視差計算を行って距離画像を生成し，距離画像に再びノイズ除去/ぼかしなどの後処理を施す，という手順で行われる．

本章では，ステレオ画像処理の中心となる距離画像生成について述べる．

2.1 距離計測

対象までの距離情報は，2 つのカメラの**視差**から求めることができる．視差とは一般には，ある計測対象となる基準点において，2 台のカメラの視線がなす角度（輻輳角）の変化量として定義される．

カメラを平行に並べた平行ステレオの場合，無限遠点が基準点となる．このため，基準となる一方のカメラ画像における，計測点の投影点に対し，もう一方のカメラ画像における同じ計測点の投影点が，カメラ画像内で何ピクセルずれているかが視差になる．

いま，左カメラを基準とする．2 つのカメラを平行に並べた場合，左カメラの中心が z 軸を通るように，また，左右カメラの並びが x 軸に平行となるように 3 次元座標を仮定する（**図 1**）．x 軸とカメラとの距離すなわちカメラの焦点距離を f，左右カメラ間の距離を B とする．また，計測点 P の左右画像上の投影点の x 座標をそれぞれ L_x, R_x とする☆．このとき，計測点 P の x 座標 P_x および z 座標 P_z に関して，

$$\begin{cases} P_x f = P_z L_x \\ (B - P_x) f = P_z (B - R_x) \end{cases}$$

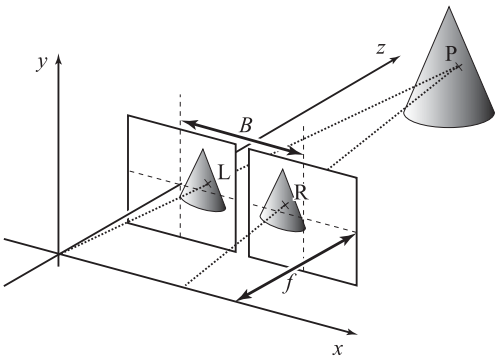

図 1 距離計測
Fig. 1 Stereo depth extraction.

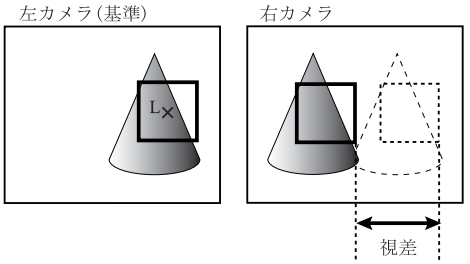

図 2 対応点と視差
Fig. 2 Correlation and disparity.

が成りたつ．よって計測点 P までの距離 P_z は，視差 $d = L_x - (R_x - B)$ を用いると，

$$P_z = \frac{Bf}{d}$$

として計算できる．

2.2 対応点探索による視差検出

以上のように，視差を求めることができれば対象までの距離を得ることができる．ただし視差を検出するためには，左右の画像中の各ピクセルの対応を正しく調べる必要がある．これは対応点探索と呼ばれ，ステレオ画像処理において最も重要な問題の 1 つである．

対応点探索の最も基本的な手法に，SSD（Sum of Squared Difference）がある．左右の画像の一方をまず基準に決める．以下では左とする．その左の画像内において，ある 1 つの点 L の周囲に小さなウィンドウを仮定する．右の画像内の点 R において，周囲に同じ大きさのウィンドウをとり，左右画像のウィンドウ内の各ピクセル値の差の 2 乗和を計算する．これを，点 R をずらしながら計算していき，その 2 乗和が最小だったときの点 R が，L の対応点として検出される（**図 2**）．なおこの際，右画像内のすべての点に対して SSD を計算する必要はない．平行ステレオの場合 L の対応点の y 座標は L の y 座標と等しく L_y と

☆ 平行ステレオの場合，投影点の y 座標はつねに同一となる．つまり視差は x 軸方向のずれとして検出され，y 座標は計測点までの距離情報には関与しない．

なるため，$y = L_y$ のスキャンライン上のみを探索すればよい．

より計算量の少ないアルゴリズムとして，SSD に代わり**SAD（Sum of Absolute Difference）**もよく用いられる．SAD はピクセル値差の 2 乗和の代わりに，ピクセル値差の総和を用いる方法である．いずれのアルゴリズムも，仮定するウィンドウの大きさを大きくとるほど正確に視差検出が行えることになるが，計算量は二次関数的に増加する．

SSD/SAD は古くから知られている手法であるが，比較的高速に計算できること，また明確なエッジが存在しない場所でも視差が検出できることから，実時間性が重要視されるロボットの視覚処理をはじめ，特に処理速度が重要となる場合には基礎的な処理の 1 つであり，今日でも一般的によく用いられている[4)~6)]．

以前は SAD 計算を高速に実行するために専用ハードウェアや DSP が用いられる場合も多かったが，近年では汎用的な PC などで計算を行うのが一般的となりつつある[7)]．そこで本稿では，SAD アルゴリズムを用いて，関数再利用によるステレオ画像処理の高速化について考える．距離計算の大部分は，このウィンドウ内のピクセル値比較が占めるため，この部分の高速化により，距離画像生成を大幅に高速化できる．

2.3 ピクセル値差の計算

本稿では，各ピクセル値に 32 bit 整数を用いる．上位 24 bit を RGB の各 8 bit に割り当て，下位 8 bit は使用しない．

視差検出のためには，ピクセル値の差を多数求める必要があることはすでに述べた．この，左右 2 つ（1 セット）のピクセル値を引数とし，ピクセル値の差を返す関数を，`pixdiff(L,R)` と定義する（**図 3**）．ウィンドウの大きさは 21×21 とした．つまり，1 度のウィンドウ比較で 441 回`pixdiff` が呼ばれる．

2.4 メディア演算命令の利用

Intel IA-32 プロセッサファミリには，MMX，SSE，SSE2 などのメディア演算命令セットがある．バイト整数の絶対差を計算する`psadbw` 命令が用意されており，SAD 計算に利用することができる．また，SPARC には V9 アーキテクチャ以降，マルチメディア拡張命令セットである VIS[8)] が実装されており，同様に`pdist` 命令を SAD 計算に利用することがきる．

以下では，SPARC の`pdist` を例に，メディア演算命令を利用した SAD 計算の高速化について述べる．

2.4.1 PDIST 命令

VIS に含まれる `pdist` 命令は，2 つの 64 bit 変数を引数とし，それぞれを 8 つの 8 bit ピクセル値と見なして，その 8 つのピクセルペアの絶対差の総和を算出する命令である．また，その結果はデスティネーションとして指定した freg（浮動小数点レジスタ）にアキュムレートされるため加算命令が省略でき，SAD 計算を高速に行うことができる（**図 4**）．

```
pixdiff(L, R){
  return(   abs(L>>24&0xff, R>>24&0xff)
          + abs(L>>16&0xff, R>>16&0xff)
          + abs(L>> 8&0xff, R>> 8&0xff) );
}
```

図 3 ピクセル値の差を計算する関数
Fig. 3 A function computes SAD.

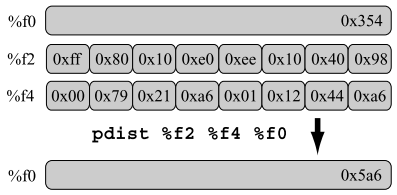

図 4 `pdist` 命令
Fig. 4 `pdist` instruction.

```
save    %sp, -112, %sp
st      %g0, [%fp-16]
st      %i0, [%fp-12]  /* 第 1 引数の転送 */
ldd     [%fp-16], %f2
st      %i1, [%fp-12]  /* 第 2 引数の転送 */
ldd     [%fp-16], %f4
fzero   %f0            /* accumulator 初期化 */
pdist   %f2, %f4, %f0  /* pdist */
std     %f0, [%fp-16]  /* 結果を転送 */
ld      [%fp-12], %i0
ret
restore
```

図 5 pdist を用いた pixdiff の記述例
Fig. 5 Sample code with `pdist`.

2.4.2 PDIST によるピクセル値差計算の実装

本稿ではピクセル値差計算に関数再利用を適用するにあたり，メディア演算命令`pdist` を用いた場合を比較対象とする．ピクセル値差計算を行う関数の中身を，アセンブリ言語レベルで`pdist` 命令を使用するようにハンドコーディングすることで実装を行った．

たとえば関数`pixdiff` の場合，引数であるピクセル値は reg（汎用レジスタ）%i0〜1 に格納されている．しかし，`pdist` は入出力に freg を使用するため，`pdist` 実行前に引数を reg から freg へ，実行後に結果を freg から reg %i0 へ転送する必要がある．SPARC では，[%fp−16]〜[%fp] の範囲をこの転送の目的で使用することができる．`pdist` を用いて`pixdiff` を記述した例を**図 5** に示す．

`pdist` は引数が double word であるため，この例

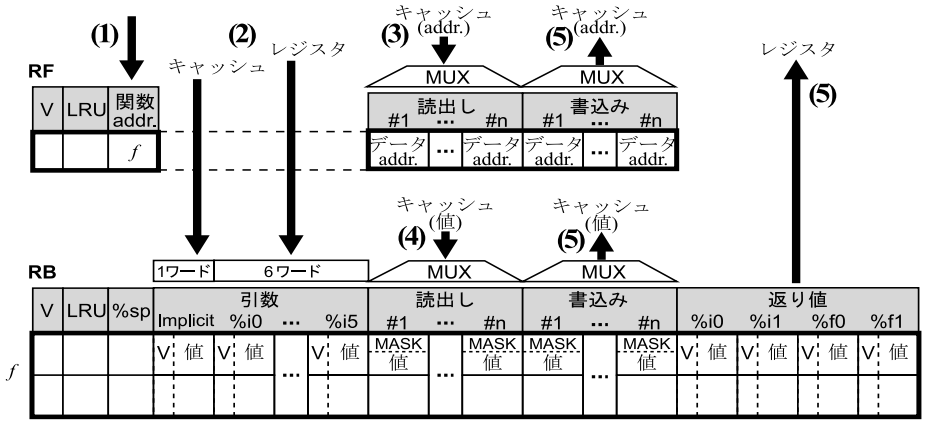

図 6 関数再利用のための表構造
Fig. 6 Structure of one-level reuse buffer.

では上位 1 word には 0 を詰めているが，最終的に結果はウィンドウ単位で総和を求めるため，2 セットのピクセル値差計算を 1 つの`pdist` で行うことができる．一般に n セットのピクセル値差の総和を計算する関数は，`pdist` で書く場合，前項で述べたようにストア命令や加算命令を必要とせず，単純に freg への 4 つの`ld` と 1 つの`pdist` の，$n/2$ 回繰返しにより記述する．

3. 区間再利用

本章では，ステレオ画像処理の高速化手法として提案する関数再利用について述べる．

区間再利用[9)~11)]（以下，再利用と略す）とは，関数呼出やループなどの命令区間において，その入力と出力の組を記憶しておき，再び同じ入力によりその命令区間が実行されようとした場合に，過去の記憶された出力を利用することで命令区間の実行自体を省略し，高速化を図る方法である．

現在，数多くの研究が行われている値予測および投機的実行[12)~14)] は，数多くの命令の投機あるいは実行結果の破棄が必要であるのに対し，再利用は実行する必要のある命令列そのものを削減できるという点で，従来とは発想の異なる高速化技術である．

今回ステレオ画像処理に適用するのは，関数呼出に対する区間再利用である，**関数再利用**である．

3.1 関数再利用

関数再利用を行うためには，関数の入出力値のペアを表に登録しておく必要がある．再び同じ関数を実行する必要が起こったとき，すでに同じ入力によりその関数が実行されている場合は，表から読み出すことで正しい出力値をただちに求めることができる．入力のセットが完全に一致していれば，実行結果は必ず同じとなり，読み出した結果を検証する必要はない．また，副次的な効果として，冗長なロード/ストア命令や消費電力を削減できることも報告されている[15),16)]．

以下，関数再利用の具体的な流れを述べる．ある関数 f を再利用するためには，f の実行時に f の入出力 f_{in}，f_{out} のみを表に登録する必要がある．関数 f を呼び出す関数を f_{p} とすると，関数 f の入力となりうるのは，大域変数および関数 f_{p} の局所変数である．よって f の入出力をメモリマップ上で識別するためには，

- 大域変数と，f の局所変数との境界
- f の局所変数と，f_{p} の局所変数との境界

を確定する必要がある．つまり，与えられた主記憶アドレスが，大域変数であるか，または，どの関数の局所変数であるかを，なんらかの方法で識別しなければならない．

我々は，SPARC ABI（Application Binary Interface）[17)] に従って記述されたプログラムに対し，SPARC ABI の規定に基づく条件を仮定することで，これら変数識別の問題を解決している．詳細に関しては文献 18) を参照されたい．

3.2 再利用機構

関数再利用を実現するためには，**関数管理表（RF）**および**入出力記録表（RB）**が必要となる．1 つの関数を再利用するために必要なハードウェア構成を**図 6** に示す．複数の関数を再利用する場合には，この構成が複数組必要となる．

各表の V は有効エントリか否かのフラグである．また LRU はそのエントリが参照された頻度を表すカウンタ値のためのフィールドで，エントリ入れ換えの際に

参照される．読み出しアドレスは RF が一括管理し，マスクおよび値は RB が管理することにより，読み出しアドレスの内容と RB の複数エントリを CAM により一度に比較する構成が可能となる．

RF は関数の先頭アドレスおよび読出し/書込みで参照される主記憶アドレスを保持している．また RB は，関数が呼び出されたときのスタックポインタの値（%sp），その関数に渡された引数，主記憶の読出し/書込みデータ，および関数の返り値を保持する．なお，各エントリ先頭の V は，そのエントリの有効性を表すフラグ，MASK はそのエントリの有効バイトを表すマスク値である．

返り値は，%i0〜1（リーフ関数の場合%o0〜1）または%f0〜1 に格納され，%f2〜3 を使用する返り値（拡張倍精度浮動小数点数）は対象プログラムには存在しないものと仮定している．

3.3 動　　作

関数 f を再利用するためには，まず f の実行時に局所変数を除外しながら，引数/返り値/大域変数，および f_{p} の局所変数に関する入出力情報を表に登録する．

引数レジスタのうち，読出しが先行したものに関しては f_{in} として，また，返り値レジスタへ書き込まれたものは f_{out} として登録する．その他のレジスタ参照に関しては，登録する必要はない．主記憶参照も同様に，読出しが先行したアドレスについては f_{in}，書込みは f_{out} として登録する．

その後，基本的には復帰命令を実行した時点で，登録中のエントリの V フィールドに有効フラグを書き込む．ただし復帰までに，他の関数の呼出し，入出力数の容量オーバー，引数の第 7 word の検出，システムコールや割込みの発生，などの擾乱が生じた場合はその時点で登録を打ち切る．

以後は f を呼び出す前に

(1) 関数先頭アドレスを検索し，RF に登録済みかどうか調べる
(2) 引数が完全に一致する RB エントリを探す
(3) 主記憶読出しデータを参照する
(4) 主記憶読出しデータの一致比較

を行い，すべての入力が一致した場合，(5) 登録されている出力，すなわち返り値，大域変数および局所変数を書き戻すことにより f の実行を省略する．

なお，3.1 節で述べたように我々は SPARC ABI の規定に基づき再利用を行っている．いま，フレームポインタを%fp とすると，SPARC ABI では，呼び出された関数 f から見た場合，第 6 word までの引数は%i0〜5 に，第 7 word 以降の引数は%fp+92 以降に入ることになる．しかしこの際，第 7 word 以降の引数の参照は%fp 相対で行われるとは限らず，f_{p} の局所変数との区別がつかない．よって簡単のため，本稿では引数が 7 word 以上の場合は再利用を行わないこととしている．

4. 関数再利用の適用

ステレオ画像処理において最も計算時間を要するピクセル値の差を求める関数に対して，再利用を適用することを考える．

背景やオブジェクトなど同じ対象の像を構成するピクセルは同じピクセル値を持つものが多く，SAD 計算関数も同じ組合せの引数で呼ばれることが多いことが予想される．すなわち，再利用を適用した場合，そのままでも効果が得られると考えられるが，この効果をさらに向上させる方法についても考える．

本章では，本稿で評価を行った，視差測定に対する関数再利用のいくつかの適用方法について述べる．

4.1 関数あたりのピクセル値差計算の多重度

視差計算の際に必要となるピクセル値の比較は，ウィンドウ単位で行われる．つまり，個々のピクセル値の差は最終的にウィンドウ単位で足しあわされたあとに比較されるので，必ずしも 1 ピクセルずつ差を計算する必要はない．

複数セットのピクセル値差の総和を計算するような関数を使用すれば，再利用表にヒットした場合に大きい効果が得られる．ただし，当然比較すべき入力が多くなるほど再利用される確率は低くなる．1 つの関数で計算するピクセル値セットの数を，以下本稿では**多重度**と呼ぶことにする．多重度 2，3 の関数`pixdiff2(`L_0`,` L_1`,` R_0`,` R_1`)` および`pixdiff3(`L_0`,` L_1`,` L_2`,` R_0`,` R_1`,` R_2`)` を定義し，それを利用した場合の効果を調べることにした．

なお，3.3 節で述べたように，再利用される関数の引数は 6 word までに限定している．ピクセル値セットをあらかじめ配列などに格納し，その配列へのポインタを関数の引数にすることで，多重度をより上げることができるが，入力一致比較が主記憶にまで及ぶためオーバヘッドが大きくなるうえ，再利用率も低下する．そこで本稿では，再利用対象となる 1 関数でまとめて計算するピクセル値差を 3 セットまでとする．これによって，引数比較はレジスタのみにとどまり，3.3 節で述べた手順 (3)，(4) は行われない．

4.2 ループオーバヘッドの削減

前項で述べたように，再利用できる関数の引数には制限があるため，再利用関数におけるピクセル値差計

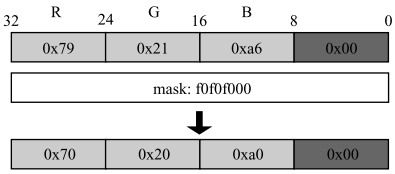

図 7 ピクセル値のマスク
Fig. 7 Masking pixel set.

算の多重度を最大で 3 とする．しかしピクセル値差比較はウィンドウ単位で行われるため，この制限がなければ最大でウィンドウ単位の 441 セットまでを 1 つの関数で計算することができる．

多重度を 21 に上げた場合のメディア演算命令を用いた実装との比較を行うため，これに対応する再利用の実装としてループオーバヘッドを削減した場合の評価を行う．

ウィンドウの大きさを 21×21 としたため，1 つのウィンドウのピクセル値総和計算には深さ 2 の 21 回ループが存在する．この内側ループを 7 つの`pixdiff3`にアンローリングしたものに関して測定し，`pdist` の $\lceil 21/2 \rceil = 11$ 回実行からなる関数に書き換えたものとの比較を行う．

4.3 曖昧再利用

一関数で計算するピクセル値差の多重度を上げた場合，再利用による効果は大きくなるが，引数が多くなるほど再利用のヒット率が低くなってしまう．

一方，JPEG 圧縮のアプリケーションにおいて，再利用部に対する入力のマッチングを厳密にしないことによって，結果の精度をほぼ保ちながら再利用率を上げることができると報告されている[19]．

そこでこの方法を適用して，ピクセル値計算の多重化による再利用率の低減を抑える．本稿では以下，この方法を**曖昧再利用**と呼ぶ．

具体的にはピクセル値において，各 RGB 表現部の下位数 bit をマスクすることでピクセル値の一致判別に寛容さを持たせる，という方法を採用した（**図 7**）．マスク値としては，RGB 値それぞれの上位 4 bit のみで一致判別を行うのに相当する f0f0f000，および上位 2 bit での判別に相当する c0c0c000 の 2 種類を用い，再利用の効果の変化および出力距離画像への影響を調べた．

5. 性能評価

5.1 評価環境

評価には，再利用機構を実装した単命令発行の SPARC-V8 シミュレータに，VIS 命令 `pdist` を追加実装したものを用いた．各パラメータを**表 1** に示す．キャッシュ構成や命令レイテンシは SPARC64-III[20] を参考にしている．

表 1 シミュレーション時のパラメータ
Table 1 Simulation parameters.

D-Cache 容量	64 KBytes	
ラインサイズ	64 Bytes	
ウェイ数	4	
Cache ミスペナルティ	20 cycles	
Register Windows 数	4 sets	
Window ミスペナルティ	20 cycles / set	
整数乗算レイテンシ	8 cycles	
整数除算 〃	70 cycles	
浮動小数点加減乗算 〃	4 cycles	
単精度浮動小数点除算 〃	16 cycles	
倍精度浮動小数点除算 〃	19 cycles	
RB（引数）⇔ レジスタ比較	1 cycle	test
RB（Read）⇔ Cache 比較	4 Bytes/cycle	test
RB（Write）⇒ Cache 書込	4 Bytes/cycle	write
RB（返り値）⇒ レジスタ書込	1 cycle	write

図 6 で示した再利用機構のサイズに関しては，RF のエントリ数を 32 としている．また，連想検索の上限，すなわち RF あたりの RB エントリ数は 512 としている．

ロードモジュールとしては，左右カメラの画像を入力とし距離画像を出力する距離画像生成プログラムを，gcc version 2.95.4（-O2 -msupersparc）によりコンパイルし，スタティックリンクにより生成したものを用いた．入力として用いたカメラ画像および生成された距離画像と，曖昧化による距離画像への影響を，**図 8** に示す．

マスクを f0f0f0 とした場合の曖昧再利用では，画質の劣化はほとんどみられない．またマスクを c0c0c0 とした場合でも劣化はごく軽微であり，物体の前後関係を判別するには支障のない画質が保たれていることが分かる．

5.2 PDIST のレイテンシ設定

UltraSPARC では`pdist` のレイテンシは，先行の`pdist` の結果にのみ依存する場合は特別に 1 となるが，他の場合は基本的に 4 である[21]．しかし，実際にはパイプラインにより一部が隠蔽されることで，4 よりも小さくなると考えられる．

ピクセル値差計算では 2.4.2 項で述べたように，2 つまたは 4 つの`ld` と 1 つの`pdist` の繰返しという使い方になる．SPARC64 GP および UltraSPARC IIe の実機を用いてこの場合のレイテンシを計測したところ，`ld` は約 1，`pdist` は SPARC64 GP で約 3，UltraSPARC IIe で約 2 という結果が得られた．

本稿で利用する SPARC-V8 シミュレータは，パイ

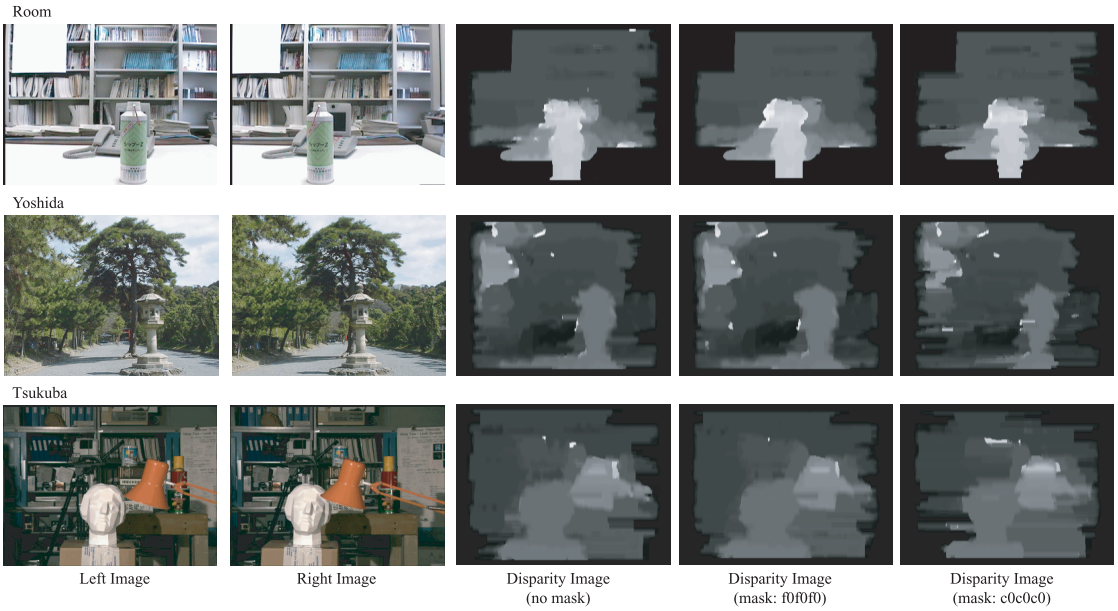

図 8 入出力画像と曖昧化による影響
Fig. 8 Stereo images and disparity images.

表 2 ld およびpdist のレイテンシ
Table 2 Latancies of ld and pdist.

	仕様上	pdist(3) (SPARC64 GP)	pdist(2) (UltraSPARC IIe)
ld	2 cycle	1 cycle	1 cycle
pdist	4 cycle	3 cycle	2 cycle

プライン部のシミュレーションに関しては実装されていないため，より実際に近い環境として，この測定結果に基づくパラメータでの計測を行うこととする．**表 2** に，シミュレーションに用いたld およびpdist のレイテンシパラメータのセットをまとめた．

5.3 測定結果

pixdiff，pixdiff2，pixdiff3，unroll という 4 つの多重度に対して，以下の各実装が距離画像生成に要したサイクル数を測定した．

- original（再利用なし）
- pdist(3)（pdist レイテンシ：3）
- pdist(2)（pdist レイテンシ：2）
- 再利用
- 曖昧再利用（mask: f0f0f0）
- 曖昧再利用（mask: c0c0c0）

pixdiff，pixdiff2，pixdiff3 はそれぞれ多重度 1〜3 の SAD 計算関数に対して，再利用を適用したものおよびpdist を用いてハンドコーディングしたものを比較している．unroll は 4.2 節で述べたように，pdist による実装は多重度を 21 にした関数で測定したものである

表 3 平均サイクル数削減率（対 original）
Table 3 Average of reduced cycles.

	pdist(3)	pdist(2)	再利用	曖昧 (f0f0f0)	曖昧 (c0c0c0)
pixdiff	37%	41%	49%	53%	54%
pixdiff2	49%	52%	49%	58%	62%
pixdiff3	53%	56%	51%	58%	66%
unroll	58%	64%	47%	51%	63%

が，再利用に関しては内側ループを 7 つのpixdiff3 にアンローリングしたものであり，再利用単位は関数pixdiff3 である．このためpdist 実装のほうが関数呼出しの回数自体が 1/7 と少なくなっていることに注意されたい．

それぞれの多重度において，再利用を適用しなかった場合に対する，pdist および関数再利用の適用による平均サイクル数削減率を**表 3** に示す．多重度を上げていった場合，通常の再利用では original に対するサイクル数削減率が低下していってしまい，pdist による実装に較べると速度向上が少ない．しかし曖昧再利用を適用することで，多重度を上げていった場合でも約 65%前後の，高いサイクル数削減率を維持できていることが分かる．

次に，多重度 1・再利用なし，の場合に要する実行サイクル数を 1 としたときの，各実装が距離画像生成に要したサイクル数の比率を，**図 9** に示す．グラフ中は左から順に，pixdiff，pixdiff2，pixdiff3，unroll の各

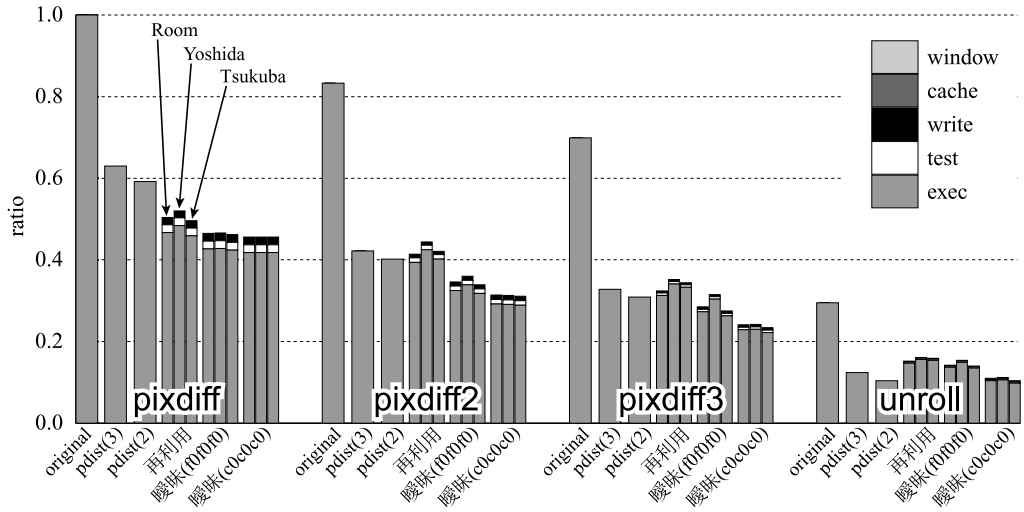

図 9 実行サイクル数の比較
Fig. 9 Executed instructions.

実装のサイクル数である．図中の凡例はサイクル数の内訳を示しており，exec は命令サイクル数，test および write は表 1 で示した，再利用表の操作に要したサイクル数である．また，cache および window はそれぞれキャッシュミスおよびレジスタウィンドウミスによるペナルティであるが，これらはグラフ中では判別できない程度の小さな値となっている．

このグラフから分かるように，曖昧再利用にループアンローリングを組み合わせた場合では，再利用率を維持しながら再利用の効果を向上させることが可能となり，多重度 1・再利用なしの実装に対して最大で 90%のサイクル数を削減することができた．メディア演算命令を用いた実装と比較しても，ループアンローリングを施さないものに関しては，曖昧化を行わない厳密な再利用でも，ほぼ同等の結果が得られている．また，ループアンローリングを適用した場合は，曖昧再利用を用いることでメディア演算命令を利用した場合とほぼ同等のサイクル数削減率が得られた．

メディア演算命令は，値の局所性の有無にかかわらず並列度の高いプログラムの高速化に適しており，一方，再利用は，並列度にかかわらず値の局所性が高いプログラムの高速化に適している．値の局所性および並列度がともに高いマルチメディア処理プログラムを用いて比較した結果，再利用によってメディア演算命令と同等の効果が得られることが分かった．

6. ハードウェア量の考察

前章まで，ステレオ画像処理の高速化という面から再利用の有効性を示し，メディア演算命令を用いた場合との比較を行ってきた．本章では，必要ハードウェア量という面において，これら再利用とメディア演算命令の比較を行う．

前章の性能評価に用いた再利用機構は，5.1 節で述べたように RF: 32，RB: 512 という大きさを設定している．本章では，この RF および RB の大きさを変化させた場合，再利用によるサイクル数削減率がどのように影響を受けるかについて述べる．またその結果から，ステレオ画像処理に曖昧再利用を適用する場合の，再利用機構に必要となるハードウェア量に関して考察し，メディア演算命令`pdist`の実現に必要なハードウェアコストとの比較を行う．

6.1 PDIST 演算

まず比較対象となる，SPARC V9 に実装されているメディア演算命令 `pdist` を実現するためのハードウェア量について考える．

UltraSPARC III には，`pdist` 命令のための FGM（Floating point/Graphics Multiply）pipeline が存在する[21]．`pdist` を実現するために必要な追加ハードウェア量を見積もるにあたり，単純に FGM に代わり`pdist` を実現する演算器を仮定し，ハードウェア量を試算することにした．

前述したように`pdist` は

- 8 bit の絶対差演算（×8）
- 絶対差（8 bit）×8 の総和演算
- 得られた結果の，freg（64 bit）へのアキュムレート

という演算からなる．

8 bit の絶対差演算（×8）を 2 段の 8 bit 減算器とセレクタで実現し，その総和演算（8 bit）および結果

のアキュムレート（64 bit に 11 bit を加算）に要する加算器，という構成を想定し，ハードウェア量を試算したところ，約 2 万 1 千トランジスタとなった[22]．

6.2 再利用機構

次に，再利用に必要となるハードウェア量を計算し，メディア演算命令との比較を行う．

関数再利用の対象となるピクセル値差計算関数は各実装につき 1 関数であるため，RF のエントリ数は 1 で十分である．実際にマスク f0f0f0，c0c0c0 に対して，RF の大きさを変化させてサイクル数計測を行い，前章で示した結果とまったく同じ結果が得られることを確認した．

次に，通常の再利用，曖昧再利用（マスク：f0f0f0），曖昧再利用（マスク：c0c0c0）のそれぞれについて，RF あたりの RB エントリ数を 8〜1024 に変化させて測定を行った．前章で使用した 3 つのステレオ画像セットによる結果にはほとんど差がみられなかったため，「Tsukuba」を用いた場合のサイクル数の変化を**図 10**，**図 11**，**図 12** に示す．

単純に引数のとりうるパターン数を考えると，たとえば`pixdiff3` と mask: c0c0c0 の組合せでは，2 bit×3（RGB）×6（引数）の 36 bit，つまり最大 $2^{36} = 64$ G 行必要ということになる．しかし図 10，図 11 の結果から，再利用および曖昧再利用（mask: f0f0f0）の場合でも 512 行の大きさの RB があれば，ほぼ最高の速度性能が得られている．また，曖昧度を高くすることで，エントリ数削減にともなう性能低下を抑えることができており，曖昧再利用（mask: c0c0c0）の場合では RB を 64 行まで縮小してもほぼ最高の速度性能，また試行中最小の 8 行でもほとんど速度低下が発生していない．

RB 1 行に要する CAM の cell 数は，入力 6 エントリ，出力 1 エントリより，$32 \times (6+1) = 224$ である．CAM 1 cell あたり 9 つのトランジスタで構成できるとし，precharge 回路なども含めて試算すると，8 エントリの RB の場合約 1 万 6 千トランジスタ，16 エントリでも約 3 万 2 千トランジスタで構成することができる．

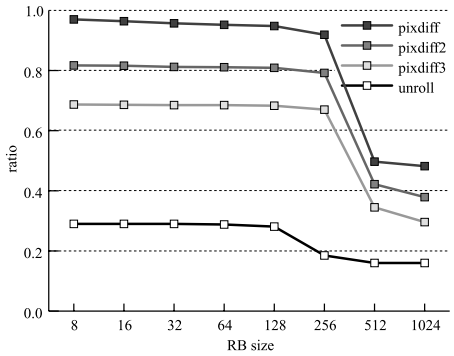

図 10 RB サイズと実行サイクル数（再利用）
Fig. 10 RB size and cycles (ordinary reuse).

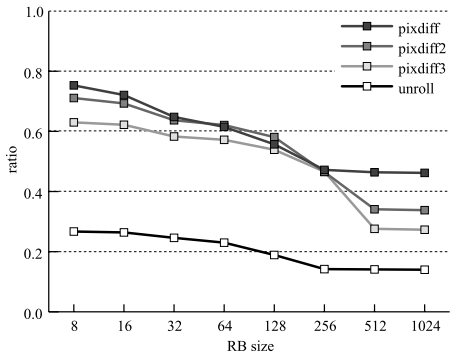

図 11 RB サイズと実行サイクル数（曖昧：f0f0f0）
Fig. 11 RB size and cycles (tolerant reuse: f0f0f0).

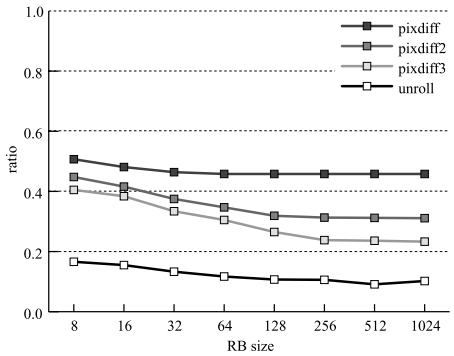

図 12 RB サイズと実行サイクル数（曖昧：c0c0c0）
Fig. 12 RB size and cycles (tolerant reuse: c0c0c0).

7. おわりに

本稿では，ステレオ画像処理において最も計算時間を要する視差測定に対し，関数再利用を適用する高速化手法を提案し，性能評価を行った．

まず，ループアンローリングを適用しない場合に関しては，曖昧化を行わない厳密な再利用でも，メディア演算命令を使用した場合と同等以上の結果が得られ，関数再利用の有効性を示した．

1 つの関数で計算するピクセル値差の多重度を上げることで，再利用しない場合に比したサイクル数削減率は 50%を下回ったが，同時に曖昧再利用を適用することで，画質を保持しつつサイクル数の削減率を約 65%まで引き上げることができた．結果，多重度を 3 まで上げかつ曖昧再利用とループアンローリングを適

用したものでは，オリジナルプログラムに対して最大90%のサイクル数を削減することができた．これはメディア演算命令を用いてループアンローリングを適用した場合とほぼ同じ性能であった．再利用は，プログラムのデータ並列度にかかわらず適用可能な手法であるが，本稿で用いた視差検出のようにメディア演算命令が最も得意とするような並列度の高いプログラムに対しても，メディア演算命令と同等の効果が得られることを示した．

また，再利用機構に必要となるハードウェア量についても考察し，曖昧再利用の場合では約64エントリのRBでほぼ最大のサイクル数削減率が得られること，約8エントリまでRBを縮小しても，ほとんど性能低下がみられないことを示した．8エントリのRBを構成するためのCAMに必要なトランジスタ数を算出したところ，1万6千トランジスタとなり，メディア演算命令を実現するためのそれと同等なトランジスタ数で実現できることを示した．

参 考 文 献

1) Sony Corp.: Entertainment Vision Sensor, http://www.sony.co.jp/SonyInfo/News/Press/200202/02-0207/ (2002).
2) Tyzx Inc.: Real-time Stereo Vision for Real-world Object Tracking, White Paper (2000).
3) 清水雄歩，中島康彦，五島正裕，森眞一郎，北村俊明，富田眞治：VLIW型メディアプロセッサを用いたステレオ画像処理の評価，情報処理学会関西支部 新・支部大会 (2002).
4) Kidono, K., Miura, J. and Shirai, Y.: Autonomous Visual Navigation of a Mobile Robot Using a Human-Guided Experience, *Robotics and Autonomous Systems*, Vol.40, No.2-3, pp.124–132 (2002).
5) Hirschmüller, H., Innocent, P.R. and Garibaldi, J.: Real-Time Correlation-Based Stereo Vision with Reduced Border Errors, *International Journal of Computer Vision*, pp.229–246 (2002).
6) Stefano, D., Marchionni, M., Mattoccia, S. and Neri, G.: A Fast Area-Based Stereo Matching Algorithm, *15th IAPR CIPPRS International Conference on Vision Interface* (2002).
7) 岡田 慧，加賀美聡，稲葉雅幸，井上博允：再帰相関法とマルチメディア命令による高速オプティカルフロー計算法，第115回コンピュータビジョンとイメージメディア研究会，pp.127–132 (1999).
8) Sun Microsystems: *The VIS™ Instruction Set*, 1.0 edition (2002).
9) Sodani, A. and Sohi, G.S.: Dynamic Instruction Reuse, *Proc. 24th International Symposium on Computer Architecture*, pp.194–205 (1997).
10) Huang, J. and Lilja, D.J.: Exploiting Basic Block Value Locality with Block Reuse, *Proc. 5th International Symposium on High-Performance Computer Architecture*, pp.106–114 (1999).
11) González, A., Tubella, J. and Molina, C.: Trace-Level Reuse, *Proc. International Conference on Parallel Processing*, pp.30–37 (1999).
12) Lipasti, M.H. and Shen, J.P.: Exceeding the Dataflow Limit via Value Prediction, *29th MICRO*, pp.226–237 (1996).
13) Wang, K. and Franklin, M.: Highly Accurate Data Value Prediction Using Hybrid Predictors, *30th MICRO*, pp.281–290 (1997).
14) Codrescu, L., Wills, D.S. and Meindl, J.: Architecture of the Atlas Chip-Multiprocessor: Dynamically Parallelizing Irregular Applications, *IEEE Trans. Comput.*, Vol.50, No.1, pp.67–82 (2001).
15) Yang, J. and Gupta, R.: Load Redundancy Removal through Instruction Reuse, *International Conference on Parallel Processing*, pp.61–68 (2000).
16) Yang, J. and Gupta, R.: Energy-Efficient Load and Store Reuse, *International Symposium on Low Power Electronics and Design*, pp.72–75 (2001).
17) Paul, R.: *SPARC Architecture, Assembly Language Programming and C*, Prentice-Hall (1999).
18) 中島康彦，津邑公暁，五島正裕，森眞一郎，富田眞治：動的命令解析に基づく多重再利用および並列事前実行，情報処理学会論文誌：コンピューティングシステム，Vol.44, No.SIG 10 (ACS 2), pp.1–16 (2003).
19) Álvarez, C., Corbal, J., Salamí, E. and Valero, M.: On the Potential of Tolerant Region Reuse for Multimedia Applications, *Proc. 15th International Conference on Supercomputing*, pp.218–228, ACM Press (2001).
20) HAL Computer Systems/Fujitsu: *SPARC64-III User's Guide* (1998).
21) Sun Microsystems: *UltraSPARC III Cu User's Manual* (2002).
22) West, N.H.E. and Eshraghian, K.: *Principles of CMOS VLSI Design: A Systems Perspective*, second edition, Addison-Wesley (1993).

(平成15年1月24日受付)
(平成15年5月 9 日採録)

津邑　公暁（正会員）

1973 年生．1996 年京都大学工学部情報工学科卒業．1998 年同大学院工学研究科情報工学専攻修士課程修了．2001 年同大学院情報学研究科博士後期課程単位取得退学．同年より同大学経済学研究科助手，現在に至る．計算機アーキテクチャ，並列処理応用，脳型情報処理に興味を持つ．人工知能学会，日本神経回路学会各会員．

清水　雄歩（学生会員）

1979 年生．2003 年京都大学工学部情報学科卒業．現在，同大学院情報学研究科通信情報システム専攻修士課程在籍．マルチメディア，ネットワーク等に興味を持つ．

中島　康彦（正会員）

1963 年生．1986 年京都大学工学部情報工学科卒業．1988 年同大学院修士課程修了．同年富士通入社．スーパコンピュータ VPP シリーズの VLIW 型 CPU，命令エミュレーション，高速 CMOS 回路設計等に関する研究開発に従事．工学博士．1999 年京都大学総合情報メディアセンター助手．同年同大学院経済学研究科助教授，現在に至る．2002 年より（兼）科学技術振興事業団さきがけ研究 21（情報基盤と利用環境）．計算機アーキテクチャに興味を持つ．IEEECS，ACM 各会員．

五島　正裕（正会員）

1968 年生．1992 年京都大学工学部情報工学科卒業．1994 年同大学院工学研究科情報工学専攻修士課程修了．同年より日本学術振興会特別研究員．1996 年京都大学大学院工学研究科情報工学専攻博士後期課程退学，同年より同大学工学部助手．1998 年同大学大学院情報学研究科助手．高性能計算機システムの研究に従事．2001 年情報処理学会山下記念研究賞，2002 年同学会論文賞受賞．IEEE 会員．

森　眞一郎（正会員）

1963 年生．1987 年熊本大学工学部電子工学科卒業．1989 年九州大学大学院総合理工学研究科情報システム学専攻修士課程修了．1992 年同大学大学院総合理工学研究科情報システム学専攻博士課程単位取得退学．同年京都大学工学部助手．1995 年同助教授．1998 年同大学大学院情報学研究科助教授．工学博士．並列/分散処理，可視化，計算機アーキテクチャの研究に従事．IEEE，ACM 各会員．

北村　俊明（正会員）

1955 年生．1978 年京都大学工学部情報工学科卒業．1983 年同大学院博士課程研究指導認定退学．同年富士通入社．汎用コンピュータ，スーパコンピュータ VPP シリーズの VLIW 型 CPU，M アーキテクチャ・命令エミュレーション，HAL 社において SPARC プロセッサ等の研究開発に従事．工学博士．2000 年京都大学総合情報メディアセンター助教授．2002 年広島市立大学情報科学部教授，現在に至る．計算機アーキテクチャに興味を持つ．電子情報通信学会，IEEE，ACM 各会員．

富田　眞治（正会員）

1945 年生．1968 年京都大学工学部電子工学科卒業．1973 年同大学院博士課程修了．工学博士．同年京都大学工学部情報工学教室助手．1978 年同助教授．1986 年九州大学大学院総合理工学研究科教授，1991 年京都大学工学部教授，1998 年同大学院情報学研究科教授，現在に至る．計算機アーキテクチャ，並列処理システム等に興味を持つ．著書「並列コンピュータ工学」(1996),「コンピュータアーキテクチャ第 2 版」(2000) 等．電子情報通信学会，IEEE，ACM 各会員．平成 7，8 年度，10，11 年度本会理事．平成 13，14 年度同関西支部長．